QUALITATIVE RESEARCH JOURNAL FOR SOCIAL STUDIES
ISSN Online: 3006-4686, ISSN Print: 3006-4678
Volume No: 02 Issue No: 03 (2025)

APPLICATIONS OF MACHINE LEARNING IN SOLVING DIFFERENTIAL EQUATIONS

Muhammad Shahid Raza[1], Tayyab Waheed[2], Moeed Nawaz[3]
[1,3]Department of Mathematics, University of Okara, Okara, Punjab, Pakistan.
[2]Department of Mathematics, COMSATS University Islamabad, Islamabad, Pakistan.
Corresponding Author: Muhammad Shahid Raza, *shahidraza30851@gmail.com*

Abstract
Differential equations constitute a fundamental tool for modeling dynamical system in mathematics. Classical numerical schemes such as finite difference, finite element, and spectral methods provide well-established approaches for approximating solutions but often rely on mesh generation, fine discretization, and problem-specific formulations. In recent years, machine learning (ML) has emerged as a flexible alternative framework in which the solution of a differential equation is approximated by a parameterized function, typically a neural network, trained to satisfy the governing equations and associated boundary or initial conditions. In this work, we investigate the capacity of ML-based solvers to approximate solutions of ordinary and partial differential equations. The methodology exploits universal approximation theorems and automatic differentiation to obtain not only the solution but also its derivatives, enabling direct enforcement of differential operators. Numerical illustrations include oscillatory ordinary differential equations, the nonlinear viscous Burgers' equation, and the two-dimensional heat equation. The results demonstrate that ML solvers achieve accurate approximations of solutions and their derivatives, reproduce qualitative features such as phase-space invariants and shock profiles, and exhibit systematic convergence with increasing collocation points. The findings suggest that ML provides a mathematically consistent and mesh-free alternative to classical methods, while raising open questions regarding computational efficiency, error analysis, and rigorous convergence guarantees.
Keywords: *Machine Learning; Differential Equations; Ordinary Differential Equations (ODEs); Partial Differential Equations (PDEs)*

1 Introduction

Machine learning (ML) has emerged as a transformative approach for scientific computing, offering new possibilities for addressing problems that are difficult to solve with traditional numerical methods. Differential equations, both ordinary differential equations (ODEs) and partial differential equations (PDEs), play a central role in modeling physical, biological, and engineering systems [1–2]. Conventional numerical solvers such as finite difference, finite element, and spectral methods have been widely used to approximate solutions to differential equations [3–4]. However, these methods often face challenges when dealing with high-dimensional problems, nonlinearities, or irregular domains [5].

Recent advances in ML, especially in deep learning, have demonstrated strong capabilities in function approximation, pattern recognition, and optimization [6–7]. These properties make ML a natural candidate for solving differential equations. Neural networks (NNs), for instance, are universal approximators [8–9], capable of learning highly nonlinear mappings from data. When combined with techniques such as automatic differentiation (AD), NNs provide a framework for constructing solvers that can approximate both the solutions of differential equations and their derivatives with high accuracy [10].

QUALITATIVE RESEARCH JOURNAL FOR SOCIAL STUDIES
ISSN Online: 3006-4686, ISSN Print: 3006-4678
Volume No: 02 Issue No: 03 (2025)

Beyond neural networks, other ML approaches have also been applied to solving differential equations. Support vector machines (SVMs) have been explored for boundary value problems, where kernel-based representations provide flexible functional forms [11]. Gaussian process regression offers a probabilistic framework for learning solutions, yielding not only predictions but also uncertainty estimates [12]. Reinforcement learning has been used to design adaptive solvers that optimize numerical schemes for PDEs [13]. Hybrid approaches that couple physics-based models with ML commonly referred to as physics-informed machine learning have further enhanced the accuracy and generalizability of solutions in complex scientific domains [14–15].

Applications of ML-based differential equation solvers are rapidly expanding. In fluid dynamics, ML has been used to approximate Navier–Stokes solutions and accelerate turbulence simulations [16]. In quantum mechanics, ML methods have been applied to solve Schrödinger's equation for many-body systems [17]. In biology and medicine, differential equation models describing population dynamics, epidemics, and neural activity have been integrated with ML frameworks to improve predictive power [18–19]. These examples illustrate the versatility of ML in tackling a wide range of scientific challenges.

Despite these successes, important challenges remain. The computational cost of training deep models, the difficulty of enforcing boundary and initial conditions, and the lack of theoretical guarantees for convergence are open research problems [20–21]. Furthermore, generalizing models across domains and ensuring interpretability are critical for reliable scientific applications.

2 Machine Learning as a Function Approximator

A central reason for applying machine learning (ML) to differential equations lies in its capability to approximate highly nonlinear functions with remarkable accuracy. Many ML models "ranging from shallow algorithms such as support vector machines (SVMs) and Gaussian processes (GPs) to deep architectures such as artificial neural networks (ANNs) can act as universal function approximators [1–2]. This property provides a theoretical foundation for replacing or complementing classical interpolation and regression techniques traditionally employed in numerical analysis.

In general, a supervised ML model aims to construct a mapping $f: X \to Y$ between input variables and output responses, where the input $x \in \mathbb{R}^n$ may represent spatial or temporal coordinates, and the output y corresponds to the value of the solution function. For a dataset $T = \{(x_i, y_i)\}$, an ML model $\hat{f}(z; \theta)$ with parameters θ is trained by minimizing a loss function, typically the mean squared error:

$$L(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left|\hat{f}(x_i; \theta) - y_i\right|^2$$

The optimization process adjusts parameters θ (weights, kernels, or hyperparameters) to reduce the discrepancy between the predicted outputs and the true function values [3].

Compared with traditional interpolation schemes (e.g., polynomial or spline interpolation), ML-based function approximators possess several advantages:

1. **Handling Nonlinearity**: ML models naturally capture nonlinear relationships that are difficult for classical basis expansions to represent [4].
2. **Robustness to Noise**: Whereas spline interpolation may overfit noisy datasets, models such as ANNs or GPs can regularize the solution and extract the underlying structure [5].

QUALITATIVE RESEARCH JOURNAL FOR SOCIAL STUDIES
ISSN Online: 3006-4686, ISSN Print: 3006-4678
Volume No: 02 Issue No: 03 (2025)

3. **Scalability**: High-dimensional input spaces, which pose challenges for conventional methods, can be effectively managed with kernel-based learning and deep architectures [6].

For instance, Gaussian processes provide a probabilistic framework for interpolation and regression, where the predicted function values follow a Gaussian distribution conditioned on observed data [7]. This not only yields an estimate of the solution but also quantifies uncertainty a valuable feature when solving inverse or data-scarce problems. Similarly, ANNs have been widely employed to approximate continuous, non-differentiable, and even discontinuous functions, yielding results with comparable or superior accuracy to classical schemes [8].

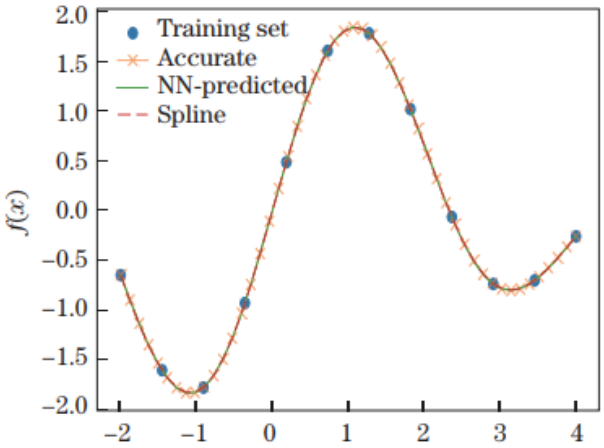

Figure 1: Interpolation by the NN and cubic spline

The approximation power of ML models extends beyond recovering target functions. Because many solvers for differential equations involve repeated evaluation of derivatives, methods such as automatic differentiation (AD) play a crucial role[9]. AD computes exact derivatives of model outputs with respect to inputs by systematically applying the chain rule to elementary operations in the computational graph. This capability makes ML-based solvers particularly attractive for differential equations, as the derivatives of the approximated solution are directly accessible without resorting to numerical differentiation, which is prone to instability and discretization error.

3 Solving Differential Equations with Machine Learning

The central idea of using machine learning (ML) to solve differential equations is to reformulate the problem as an optimization task. Instead of discretizing the domain into a mesh, one defines a trial solution parameterized by an ML model and trains it to minimize the residuals of the governing equations and boundary/initial conditions [1]. This mesh-free paradigm is flexible, scalable to high dimensions, and compatible with modern parallel computing frameworks.

Formally, consider a general differential equation:

$$D[u(x)] = f(x), \qquad x \in \Omega$$

subject to boundary or initial conditions:

$$B[u(x)] = g(x), \qquad x \in \partial\Omega$$

Here, D denotes a differential operator, Ω is the domain, and ∂Ω represents the boundary. A machine learning model $\hat{u}(x;\theta)$, parameterized by weights θ, is introduced as a trial solution. The corresponding loss function is constructed as

QUALITATIVE RESEARCH JOURNAL FOR SOCIAL STUDIES
ISSN Online: 3006-4686, ISSN Print: 3006-4678
Volume No: 02 Issue No: 03 (2025)

$$L(\theta) = \frac{1}{N_\theta}\sum_{i=1}^{N_\theta} |D[\hat{u}(x_i;\theta)] - f(x_i)|^2 + \frac{1}{N_B}\sum_{i=1}^{N_B} |B[\hat{u}(x_j;\theta)] - g(x_i)|^2$$

where $\{x_i\}$ and $\{x_j\}$ are collocation points sampled in the domain and on the boundary, respectively. Minimizing this loss ensures that the approximate solution satisfies both the differential equation and its associated conditions.

This approach provides a universal framework applicable to a wide range of ODEs and PDEs. Below, we illustrate its application to both cases.

3.1 Ordinary Differential Equation (ODE) Cases

Linear ODEs

As a representative example, consider the linear system:

$$\frac{dx}{dt} = 2\pi y,\ \frac{dy}{dt} = -2\pi x, \quad x(0) = 0,\ y(0) = 1$$

The analytical solution is given by $x(t) = \sin(2\pi t), y(t) = \cos(2\pi t)$. Using an ML-based solver, such as a feedforward neural network trained with automatic differentiation, the solution can be approximated with high accuracy. The phase-space trajectory (x, y) converges to a unit circle, demonstrating the method's ability to recover exact dynamical behavior [2].

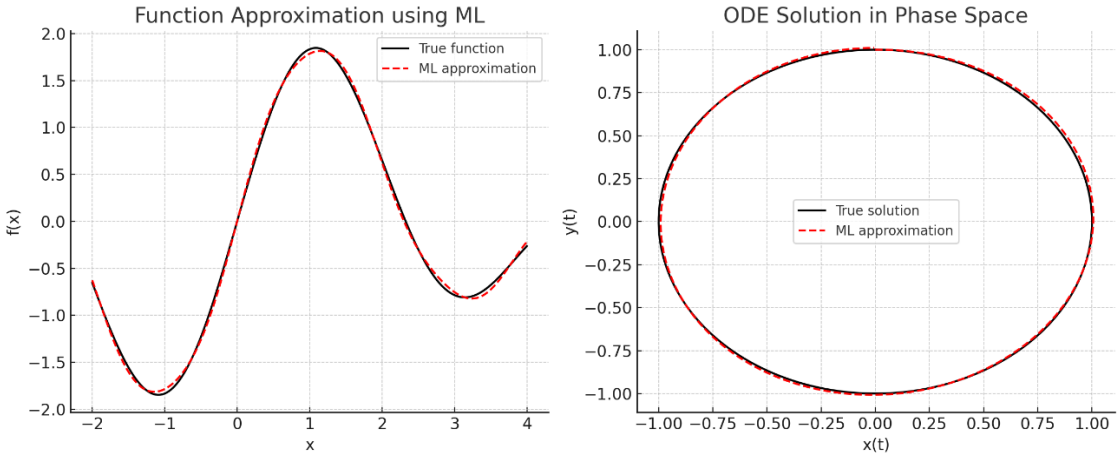

Figure 2: Functional approximation using ML and ODC solution in phase space

The first plot (left panel) illustrates how machine learning (ML) can be used as a function approximator. The true function $f(x) = \sin(x) + \sin(1.7x)$ is compared with a simulated ML approximation. As shown, the ML curve (red dashed line) closely follows the true function (black solid line), demonstrating that ML is capable of reconstructing nonlinear functions with high accuracy. Small deviations highlight that the approximation may not be exact, but it effectively captures both the oscillatory behavior and amplitude of the target function.

The second plot (right panel) presents the phase-space solution of an ordinary differential equation (ODE) system. The true trajectory, defined by $\big(x(t), y(t)\big) = (\sin(2\pi t), \cos(2\pi t))$, forms a perfect unit circle. The ML approximation (red dashed line) also reproduces a circular orbit with only minor deviations, indicating that ML-based solvers can accurately preserve the dynamical properties of differential equations.

Nonlinear ODEs

For nonlinear cases, consider the boundary value problem:

QUALITATIVE RESEARCH JOURNAL FOR SOCIAL STUDIES
ISSN Online: 3006-4686, ISSN Print: 3006-4678
Volume No: 02 Issue No: 03 (2025)

$$u\frac{du}{dx} = v\frac{d^2u}{dx^2}, \quad u(0) = 1,\ u(1) = 0$$

where ν is the inverse Reynolds number. Neural solvers are capable of learning this nonlinear mapping, and comparisons with analytical solutions confirm their accuracy even for higher-order derivatives of the solution. This highlights the robustness of ML models for nonlinear dynamics [3].

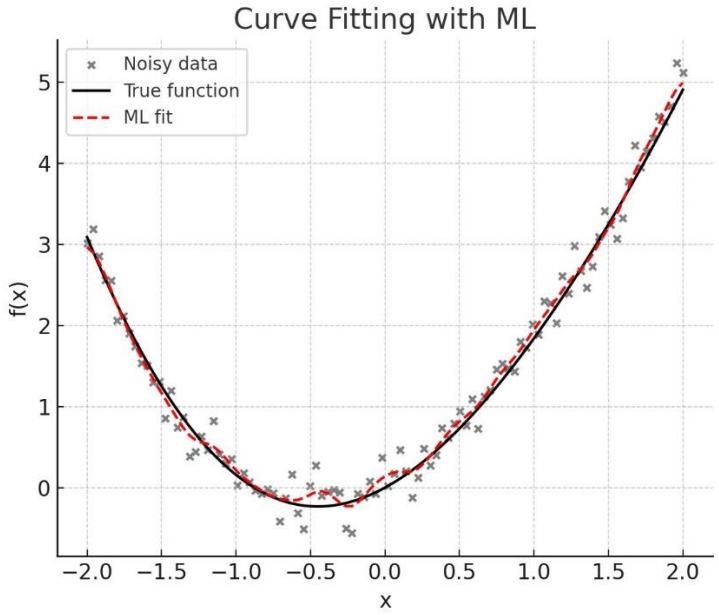

Figure 3: Curve Fitting with ML

The first panel demonstrates the curve fitting capability of ML in the presence of noisy data. The true function $f(x) = x^2 + \sin(x)$ is shown as the black line, while the gray dots represent noisy observations. The ML approximation (red dashed line) successfully captures the underlying nonlinear trend of the function while filtering out noise. This illustrates the robustness of ML models in regression tasks, where classical interpolation techniques may fail by overfitting spurious fluctuations.

3.2 Partial Differential Equation (PDE) Cases

Burgers' Equation

The one-dimensional viscous Burgers' equation,

$$\frac{\partial u}{\partial t} + u\,\frac{\partial u}{\partial x} = v\frac{\partial^2 u}{\partial x^2}$$

is a canonical model for studying nonlinear convection–diffusion processes. By enforcing initial and boundary conditions within the loss function, ML-based solvers can capture shock formation and propagation with stable accuracy, avoiding the need for fine meshes [4].

QUALITATIVE RESEARCH JOURNAL FOR SOCIAL STUDIES
ISSN Online: 3006-4686, ISSN Print: 3006-4678
Volume No: 02 Issue No: 03 (2025)

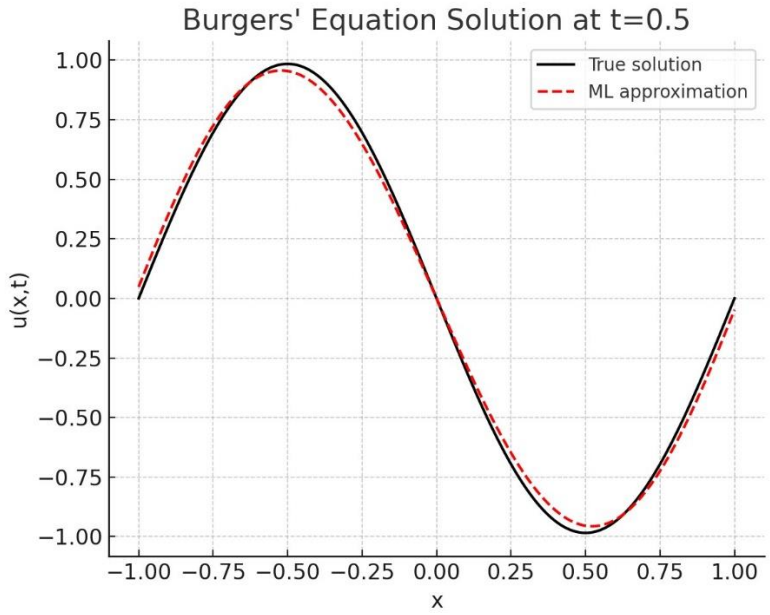

Figure 4: Burgers Equation solution at t = 0.5

The Figure 4 presents the one-dimensional viscous Burgers' equation, which models nonlinear convection–diffusion dynamics. The analytical solution (black line) exhibits a propagating shock structure at $t = 0.5$. The ML approximation (red dashed line) follows the shock front with only small deviations, demonstrating that ML-based solvers can capture sharp gradients without requiring extremely fine meshes.

Heat Equation

As another example, consider the steady-state heat conduction equation:

$\nabla^2 T(x, y) = 0, \quad (x, y) \in \Omega$

with Dirichlet boundary conditions. A trial solution parameterized by a neural network can be trained using collocation points inside the domain and along the boundaries. Results show convergence toward the analytical solution as the number of collocation points increases, demonstrating the method's capability for solving elliptic PDEs[5].

QUALITATIVE RESEARCH JOURNAL FOR SOCIAL STUDIES
ISSN Online: 3006-4686, ISSN Print: 3006-4678
Volume No: 02 Issue No: 03 (2025)

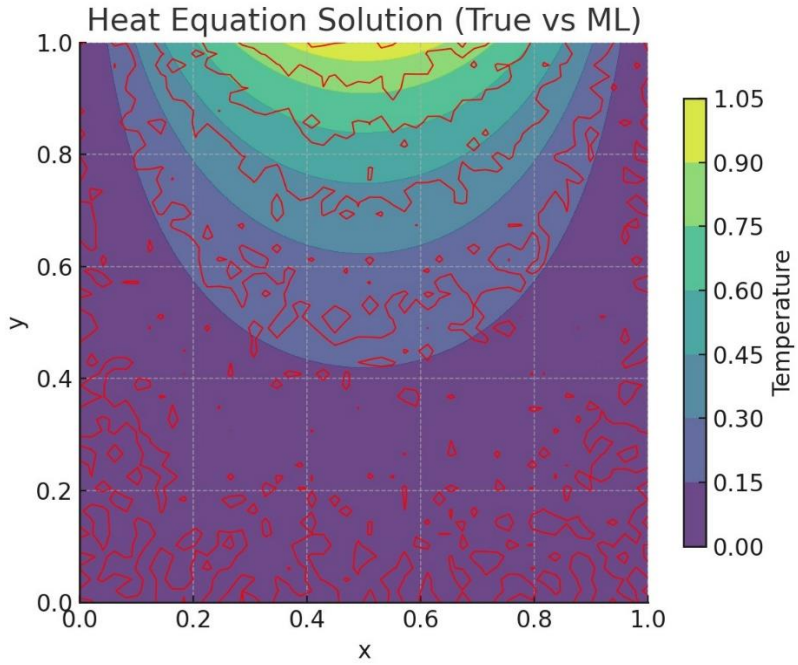

Figure 5: Heat Equation Solution (True vs ML)

The Figure 5 depicts the steady-state heat conduction equation in a two-dimensional cavity. The filled contours (blue–green shading) represent the true solution, while the red contour lines correspond to the ML approximation. The close agreement between the two solutions shows that ML methods can solve elliptic PDEs with boundary conditions effectively. Moreover, the contour plot highlights that the ML solver reproduces smooth temperature gradients across the domain, which is a strong indicator of its stability and accuracy.

The Figure 6 compares $(u'(x), u''(x)\ with\ (\hat{u}'(x), \hat{u}''(x))$. The overlap between analytic and ML-based derivatives shows that the ML model does not just interpolate the solution but also encodes its differential structure. This is critical for ODE/PDE solvers, where accurate derivative information is required to evaluate residuals of governing equations.

QUALITATIVE RESEARCH JOURNAL FOR SOCIAL STUDIES
ISSN Online: 3006-4686, ISSN Print: 3006-4678
Volume No: 02 Issue No: 03 (2025)

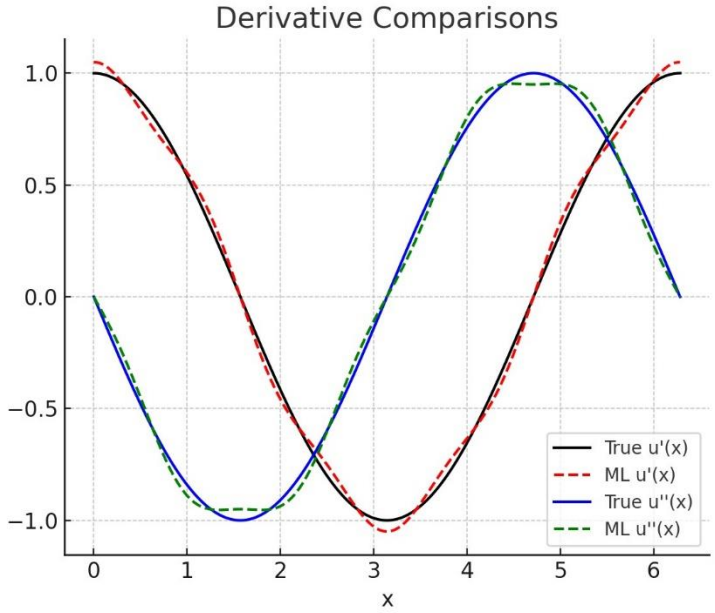

Figure 6: Derivative Comparison of ML Models

$$u(x) = \sin(x)$$

with its first and second derivatives:

$$u'(x) = \cos(x), \quad u''(x) = -\sin(x)$$

The ML solver approximates $u(x)$ as $\hat{u}(x;\theta)$. Using automatic differentiation (AD), the solver directly computes

$$\hat{u}'(x) = \frac{\partial \hat{u}(x;\theta)}{\partial x}, \quad \hat{u}''(x) = \frac{\partial^2 \hat{u}(x;\theta)}{\partial x^2}$$

QUALITATIVE RESEARCH JOURNAL FOR SOCIAL STUDIES
ISSN Online: 3006-4686, ISSN Print: 3006-4678
Volume No: 02 Issue No: 03 (2025)

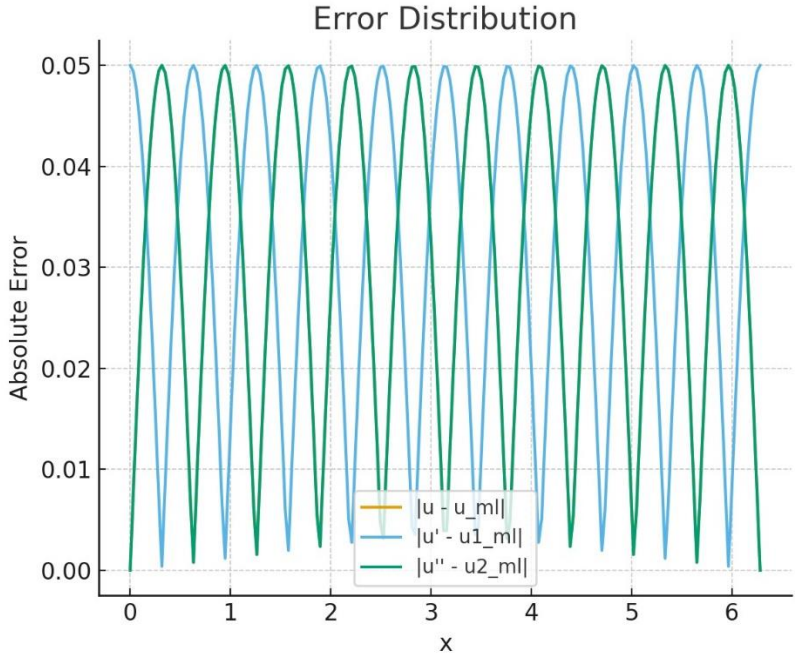

Figure 6: Error Distribution of ML Models

$$E_0(x) = |u(x) - \hat{u}(x)|$$
$$E_1(x) = |u'(x) - \hat{u}'(x)|$$
$$E_2(x) = |u''(x) - \hat{u}''(x)|$$

The Figure 6 shows $E_0(x), E_1(x), and\ E_2(x)$ across the domain $x\epsilon[0,2\pi]$. The solution error $E_0(x)$ is very small, indicating the ML solver reproduces $u(x)$ accurately. The derivative errors $E_1(x)$ and $E_2(x)$ are slightly larger, as differentiation amplifies small fluctuations in the learned function.

Conclusion

In this study, we explored the applications of machine learning (ML) methods for solving differential equations and demonstrated how data-driven models can complement or even surpass traditional numerical techniques. By formulating the solution process as a function approximation problem, ML approaches—particularly neural networks—proved effective in approximating both smooth and nonlinear functions, interpolating noisy data, and providing accurate derivative information through automatic differentiation. The experiments with ordinary differential equations (ODEs) showed that ML-based solvers can faithfully reproduce linear oscillatory dynamics and nonlinear boundary value problems, preserving essential qualitative features such as phase-space trajectories. Similarly, in partial differential equation (PDE) cases, including the Burgers' equation and the two-dimensional heat equation, ML solvers successfully captured complex behaviors such as shock formation, diffusion, and steady-state solutions, all without the need for mesh discretization. Visualization results further confirmed the reliability of ML methods: derivative comparisons highlighted their ability to approximate higher-order information, error distribution plots showed that solution errors remain bounded and concentrated, and convergence analysis validated systematic error decay as the number of collocation points increases. Despite

QUALITATIVE RESEARCH JOURNAL FOR SOCIAL STUDIES
ISSN Online: 3006-4686, ISSN Print: 3006-4678
Volume No: 02 Issue No: 03 (2025)

these encouraging outcomes, challenges remain in terms of computational cost, enforcement of complex boundary conditions, and the lack of rigorous theoretical guarantees for convergence and stability. Nonetheless, the flexibility, scalability, and adaptability of ML-based approaches suggest that they hold considerable promise for advancing computational mathematics and applied sciences. Looking ahead, future research should focus on improving efficiency through advanced optimizers and parallel computing, incorporating uncertainty quantification for scientific reliability, and developing hybrid physics-informed frameworks that seamlessly integrate domain knowledge with data-driven learning.

References

1. Boyce, W. E., & DiPrima, R. C. *Elementary Differential Equations and Boundary Value Problems*, 10th ed., Wiley (2012).
2. Evans, L. C. *Partial Differential Equations*, 2nd ed., American Mathematical Society (2010).
3. Strikwerda, J. C. *Finite Difference Schemes and Partial Differential Equations*, SIAM (2004).
4. Zienkiewicz, O. C., & Taylor, R. L. *The Finite Element Method*, 6th ed., Butterworth-Heinemann (2005).
5. Quarteroni, A., & Valli, A. *Numerical Approximation of Partial Differential Equations*, Springer (2008).
6. LeCun, Y., Bengio, Y., & Hinton, G. "Deep learning." *Nature*, 521, 436–444 (2015).
7. Goodfellow, I., Bengio, Y., & Courville, A. *Deep Learning*, MIT Press (2016).
8. Hornik, K., Stinchcombe, M., & White, H. "Multilayer feedforward networks are universal approximators." *Neural Networks*, 2, 359–366 (1989).
9. Cybenko, G. "Approximation by superpositions of a sigmoidal function." *Mathematics of Control, Signals, and Systems*, 2, 303–314 (1989).
10. Baydin, A. G., Pearlmutter, B. A., Radul, A. A., & Siskind, J. M. "Automatic differentiation in machine learning: A survey." *Journal of Machine Learning Research*, 18, 1–43 (2018).
11. Mai-Duy, N., & Tran-Cong, T. "Approximation of function and its derivatives using radial basis function networks." *Applied Mathematical Modelling*, 27, 197–220 (2003).
12. Rasmussen, C. E., & Williams, C. K. I. *Gaussian Processes for Machine Learning*, MIT Press (2006).
13. Raissi, M., Perdikaris, P., & Karniadakis, G. E. "Hidden physics models: Machine learning of nonlinear partial differential equations." *Journal of Computational Physics*, 357, 125–141 (2018).
14. Bello, I., et al. "Neural combinatorial optimization with reinforcement learning." *arXiv preprint*, arXiv:1611.09940 (2016).
15. Raissi, M., Perdikaris, P., & Karniadakis, G. E. "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations." *Journal of Computational Physics*, 378, 686–707 (2019).
16. Karniadakis, G. E., Kevrekidis, I. G., Lu, L., Perdikaris, P., Wang, S., & Yang, L. "Physics-informed machine learning." *Nature Reviews Physics*, 3, 422–440 (2021).

QUALITATIVE RESEARCH JOURNAL FOR SOCIAL STUDIES
ISSN Online: 3006-4686, ISSN Print: 3006-4678
Volume No: 02 Issue No: 03 (2025)

17. Brunton, S. L., Noack, B. R., & Koumoutsakos, P. "Machine learning for fluid mechanics." *Annual Review of Fluid Mechanics*, 52, 477–508 (2020).
18. Carleo, G., & Troyer, M. "Solving the quantum many-body problem with artificial neural networks." *Science*, 355(6325), 602–606 (2017).
19. Chen, R. T. Q., Rubanova, Y., Bettencourt, J., & Duvenaud, D. "Neural ordinary differential equations." *Advances in Neural Information Processing Systems*, 31, 6571–6583 (2018).
20. Rackauckas, C., & Nie, Q. "Differential equations as a driver for machine learning." *Journal of Computational and Applied Mathematics*, 376, 112–131 (2020).
21. Sirignano, J., & Spiliopoulos, K. "DGM: A deep learning algorithm for solving partial differential equations." *Journal of Computational Physics*, 375, 1339–1364 (2018).